

Distributed Perception Networks: An Architecture for Information Fusion Systems Based on Causal Probabilistic Models

Gregor Pavlin, Patrick de Oude, Marinus Maris

Intelligent Autonomous Systems Group
Informatics Institute, University of Amsterdam
gpavlin@science.uva.nl

Thomas Hood

Thales Research and Technology

Abstract—We introduce Distributed perception networks (DPNs), a distributed architecture for efficient and reliable fusion of large quantities of heterogeneous and noisy information. DPNs consist of agents, processing nodes with limited fusion capabilities, which cooperate and can autonomously form arbitrarily large compound classifiers. DPNs make use of causal models, which often facilitate analysis, design and maintenance of complex information fusion systems. Namely, observations obtained from different information sources often result from causal processes which in turn can be modeled with relatively simple, yet mathematically rigorous and compact probabilistic causal models. Such models, in turn, facilitate decentralized world modeling and information fusion.

I. INTRODUCTION

Contemporary situation assessment and control applications increasingly require efficient and robust interpretation of large amounts of heterogeneous information. Such interpretation requires fusion systems that can cope with imprecise domain models and that can process large amounts of noisy and imprecise or partially inaccurate information coming from different sources via different media. Building such fusion systems can be very challenging. However, in this paper we show that, by considering causal domain models, powerful fusion systems can be constructed.

Namely, as it was pointed out in [6], humans often tend to reason in terms of casual processes. While it is often difficult to precisely formulate causality, it seems that causal models facilitate identification of dependencies among different facts in the domain. This is very relevant for a broad class of real world domains, where each fact/event is significantly influenced only by a small fraction of other events. Thus, by considering such locality of causal relations, we can often construct models that ignore a significant portion of events which do not influence facts we are interested in. In other words, by considering causal relations we can obtain compact models which at the same time describe the domain sufficiently well with respect to the given problem.

Moreover, in real world domains we are usually confronted with stochastic causal processes, which can be modeled through Causal Bayesian networks (CBN). Bayesian networks (BN) in general are probabilistic models which describe statistical relationships among different phenomena.

CBNs are a special class of BNs where the relationships between variables are interpreted as causal.

By explicitly considering probabilistic causal models, we introduce Distributed Perception Networks (DPN), multi-agent systems (MAS) architecture that supports efficient and reliable fusion of large quantities of heterogeneous and noisy information. DPNs consist of agents, processing nodes with limited fusion capabilities, which cooperate and can autonomously form arbitrarily large distributed estimators.

Each agent has a limited domain expertise encoded through a local CBN. CBNs provide a theoretically rigorous and compact mapping between hidden events of interest and observable events to which the latter are causally related.

For each hypothesis, domain of interest and a particular constellation of information sources, a specific distributed domain model is required. DPN agents, which represent basic building modeling blocks, can organize into an appropriate DPN, such that observations from various available sources (sensors, database systems, GSM networks and the WWW) are embedded in a distributed causal model which allows efficient and theoretically rigorous mapping between the hypotheses of interest and observations.

A system of DPN agents can autonomously form organizations which correspond to correct distributed causal models. The self-configuration process is based on local CBNs and exploits the locality of causal relations; new variables and relations are added to an initial CBN in such a way that the relations between the variables in the initial CBN do not change.

Through self-configuration a network of DPN agents is constructed in a distributed manner. Given a request for information about a particular proposition, a DPN agent that has a local domain model for that proposition searches for other relevant agents that can serve as information sources. In this way agents assemble local models into distributed total domain models at runtime. By doing it on the fly, DPNs can cope with changing constellations of information sources, a feature that is very advantageous in real world applications. In addition, by physically spreading the model instantiation over a network of DPN agents it is possible to overcome processing and communication bottlenecks and to eliminate single points of failure. In addition, DPN approach facilitates

maintenance and design of complex fusion systems.

The paper is organized as follows. We first discuss implications of causal models in the context of real world information fusion problems. We explain why causal models can describe the targeted domains and show how they facilitate the design of distributed fusion systems. By considering causal processes we introduce a set of modeling constraints and rules which facilitate design of very robust, self organizing fusion systems. Note that for the sake of clarity, we keep examples rather simple and we present simplified versions of modeling components and belief propagation algorithms.

II. STATE ESTIMATION WITH CAUSAL BAYESIAN NETWORKS

In this paper we focus on domains where situation assessment requires estimation of states which cannot be observed directly, but we can observe their effects (i.e. symptoms). We assume that situations can be described through states of discrete random variables, as for example binary variables with the states representing the presence/absence of fire and smoke. Moreover, we assume that hidden and observed states in the domain, such as the presence of smoke or report from a smoke detector, materialize through 'sampling' from some true distribution over the combinations of possible states. Thus, in a particular situation certain states materialized while others did not, which corresponds to a point mass distribution over the possible states of discrete random variables in the targeted domain. Consequently, the state estimation can be reduced to classification of the possible combinations of relevant states; e.g. we classify a set of observations either as an instance of class $Fire = true$ or $Fire = false$.

In addition, in a significant class of domains sequences of observations result from stochastic causal processes, which in turn can be described through probabilistic causal models. Such models basically capture dependencies, such as the fact that materialization of certain states makes realization of other states more/less probable.

In other words, given a causal process that produces certain observable states, we can formulate information fusion as a mapping from different observations to hidden causes. This requires domain models which explicitly describe relations between every observable event and their hidden causes. In order to be able to cope with inherent uncertainties, we use causal Bayesian networks, probabilistic models which can describe stochastic causal processes in a rigorous and compact way [5].

A. Causal Bayesian Networks

A Bayesian network is defined as a tuple $\langle \mathcal{D}, P \rangle$, where $\mathcal{D} = \langle \mathcal{V}, E \rangle$ is a directed a-cyclic graph defining a domain $\mathcal{V} = \{V_1, \dots, V_n\}$ and a set of directed edges $\langle V_i, V_j \rangle \in E$ over the domain. The joint probability distribution over the domain \mathcal{V} is defined as

$$P(\mathcal{V}) = \prod_{V_i \in \mathcal{V}} P(V_i | \pi(V_i)),$$

where $P(V_i | \pi(V_i))$ is the conditional probability table (CPT) for node V_i given its parents $\pi(V_i)$ in the graph. In this paper, we assume that each node represents a discrete variable.

BNs can be used as causal models that describe probabilistic relations between different hidden phenomena and heterogeneous sensory observations (see example in figure 1). In a BN we choose a hypothesis node H with states h_i and compute probability distribution $P(H | \mathcal{E})$ over H for a given evidence pattern \mathcal{E} (e.g. sensory observations). Evidence \mathcal{E} corresponds to a certain constellation of node instantiations and subsequent inference (i.e. information fusion) results in a distribution $P(H | \mathcal{E})$ that determines a "score" $P(h_i | \mathcal{E})$ for each hypothesis $h_i \in H$.

B. Causal Domain Models: an Example

By using probabilistic causal models we can represent different observations and their causes in a theoretically rigorous and compact way. It turns out that CBNs are a relevant tool for a significant class of information fusion systems, where we can identify causal relationships between hidden events and events that can be observed¹.

We illustrate this with the help of a simple example. Let's assume a fire detection system that makes use of different types of sensors, such as smoke detectors, thermometers and cameras detecting flames. Each sensor measures a particular physical quantity. For example, smoke detectors could measure conductivity of the ionized air in their vicinity. Electronic circuits evaluate air conductivity and generate streams of sensor reports. Moreover, the j -th report from the i -th sensor is represented by a random variable R_j^i . A report signaling the presence or absence of some phenomenon, such as smoke, is characterized by $R_j^i = true$ or $R_j^i = false$, respectively. The smoke concentration exceeding some threshold for a certain period of time (i.e. a time slice) would cause a change of the air conductivity, such that the electronic circuit would measure a high conductivity within that time slice. In such a case the sensor is considered to work properly if $P(R_j^i = true)$, the probability that a report from this sensor will indicate the presence of smoke, is greater than $P(R_j^i = false)$, the probability that a report will signal the absence of smoke, i.e. $P(R_j^i = true) > P(R_j^i = false)$. If the smoke were absent, a sensor that works correctly would generate reports for which the distribution $P(R_j^i)$ would satisfy relation $P(R_j^i = true) < P(R_j^i = false)$. However, the probability distribution $P(R_j^i)$ does not depend only on the presence of the phenomenon that we are trying to infer. For example, the sensor electronics might fail, the sensor could be covered by ice, low temperatures could influence the measurement of the conductivity, etc. In other words, the distribution $P(R_j^i)$ over reports depends on many different factors which are often not well known.

In order to avoid detailed modeling of the processes resulting in sensor reports, we introduce the *sensor propensity* concept that summarizes two types (classes) of situations characterized by combinations of the states of the electronic

¹In this paper an event is synonymous to a realization of certain states

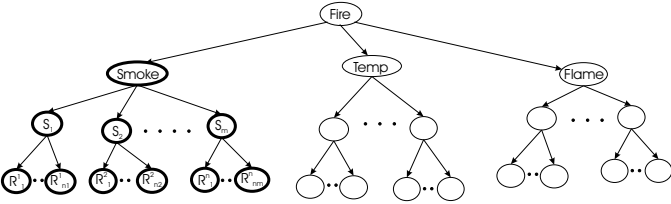


Fig. 1. A causal model capturing relations between the hidden phenomenon *Fire* and different types of sensor observations. Highlighted nodes represent a model fragment that captures causal relations between *Smoke*, sensor contexts S_i and sensor reports R_j^i .

components and the states of the sensor's vicinity (e.g. conductivity of the ionized air). We represent the *sensor propensity* by a binary variable S . $S = true$ denotes the class of state combinations which influences the sensing process in such a way that the probability of obtaining a sensor report confirming some phenomenon is greater than 0.5; i.e. $P(R_j^i = true|S = true) > P(R_j^i = false|S = true)$. The class of situations corresponding to $S = false$ would bias the distribution over sensor reports $P(R|S)$, such that $P(R_j^i = true|S = false) < P(R_j^i = false|S = false)$.

By representing sensor propensity by a binary node we can use simple causal models to describe the relations between the hidden phenomenon, sensor propensity and the observation sequences. For example, for smoke detectors such relations are captured by highlighted nodes in figure 1, where node *Smoke* represents the presence/absence of smoke, binary nodes S_i represent propensity of the i -th sensor, while nodes R_j^i correspond to sensor reports obtained from the i -th sensor during a particular time slice.

With each sensor we introduce an independent partial causal process that is initiated through some hidden phenomenon. For example, by introducing a new smoke detector the presence of smoke will initiate different processes in the sensor's circuitry which will eventually produce sensor reports. Consequently, in a BN, modeling such causal processes, each sensor corresponds to a network fragment which is conditionally independent of other network components given the node modeling the phenomenon. Thus, by using many heterogeneous information sources we can obtain BN topologies with many independent fragments (see figure 1).

Moreover, we assume domains where hidden states of the environment do not change during a single time slice. For example, in figure 1 nodes *Fire* and *Smoke* capture the presence/absence of phenomena. We assume that fire is present throughout a time slice; i.e. $Fire = true$ for the duration of the time slice. We call such phenomena quasi-static. On the other hand, within a finite time slice, we obtain sequences of observations which are influenced by the hidden phenomena represented through latent variables in a causal model. Such observations are represented through leaf nodes. In contrast to hidden variables, each observation corresponds to a particular time instant. For a significant class of sensors, we can assume that sequences of observations result from

first order Markov processes. However, it was shown that such a process can be modeled as a set of branches rooted in a single node, if the hidden phenomenon is quasi static (see [1]). In other words, for each observation from a sequence obtained within a single time slice, a new leaf node is appended to a common root node.

C. Domain Models and Factorization

The topology of causal models is reflected in the factorization of the distributions of interest. As we will show later, efficient distribution of inference processes depends on the factorization corresponding to a particular causal process.

Given a hypothesis variable H we can define a *conditionally independent network fragment*:

Definition 1: Given a BN and a classification variable H , i^{th} *conditionally independent network fragment* \mathcal{F}_i^H is a set of nodes that include node H and are d-separated from other parts of a BN by H . All nodes within \mathcal{F}_i^H are dependent given the variable H and there always exists a path between any two nodes.

D-separation implies conditional independence between the modeled variables, which corresponds to a specific factorization of the posterior probability over $\hat{P}(H|\mathcal{E})$. Namely, $\bigcap_i \mathcal{F}_i^H = \{H\}$, which means that the potentials corresponding to a particular network fragment \mathcal{F}_i^H do not share any variables with the potentials associated with other network fragments, except the hypothesis variable H . Thus, each network fragment \mathcal{F}_i^H is associated with a factor $\phi_i(H)$ resulting from a marginalization of all variables from this fragment except H .

Let's consider again the causal model depicted in figure 1 and assume that we try to determine the joint probability distribution $P(Fire, \mathcal{E})$ over the variables *Fire* and all observations \mathcal{E} collected during a fusion time slice. The DAG structure implies a certain factorization of $P(Fire, \mathcal{E})$, where each factor corresponds to different conditionally independent network fragments given the hypothesis node *Fire*:

$$P(Fire, \mathcal{E}) = P(Fire)P(\mathcal{E}_{smoke}|Fire)P(\mathcal{E}_{temp}|Fire) \cdot P(\mathcal{E}_{flame}|Fire) \quad (1)$$

where \mathcal{E}_i denote sets of instantiated states in the network fragments modeling Smoke, Temperature, etc. Each of these factors can be computed through a sequence of multiplications and marginalizations, independently of other factors. For example, factor $P(Fire|\mathcal{E}_{smoke})$ can be computed as:

$$P(\mathcal{E}_{smoke}|Fire) = \sum_{Smoke} [P(Smoke|Fire) \cdot P(\mathcal{E}_{S_1}|Smoke)P(\mathcal{E}_{S_m}|Smoke)], \quad (2)$$

where \mathcal{E}_{S_i} denotes the set of instantiated states in the fragment modeling the propensity of the i^{th} sensor measuring the ionized air conductivity. In this expression we can again identify conditionally independent factors $P(\mathcal{E}_{S_i}|Smoke)$,

each corresponding to a single sensor. Each $P(\mathcal{E}_{S_i}|Smoke)$ can be computed independently of other factors:

$$P(\mathcal{E}_{S_i}|Smoke) = \sum_{S_{coni}} P(S_i|Smoke) \prod_j P(R_j^i|S_i). \quad (3)$$

With the help of these factorizations we can infer the posterior probability for the presence of *Fire* as follows: $P(Fire|\mathcal{E}) = P(Fire, \mathcal{E})/P(\mathcal{E})$.

D. Fusion Task Decomposition

A fusion process can be viewed as a sequence of multiplications and summations. The factorization example from the previous section illustrates that certain factors can be computed independently of other factors. Thus, we can consider the computation of a certain factor a partial fusion process or fusion task. Consequently, if a causal process features conditional independencies, complex fusion tasks can be accomplished through hierarchies of simpler fusion tasks, each corresponding to a partial estimation problem focusing on a particular hidden event from a causal chain.

III. DISTRIBUTED PERCEPTION NETWORKS

The factorization properties illustrated in section II-C suggest that conditional independence in causal models can be used to identify independent network fragments which correspond to independent factors. In other words, each factor can be computed independently of other factors, which implies that we can easily distribute probabilistic models and belief propagation among different processing units.

By considering this property, we introduce Distributed Perception Networks (DPN), multi agent systems for efficient and robust fusion.

A. DPN Agents: Basic Modeling Building Blocks

A DPN agent is an information processing unit, a basic modeling building block that supports self organization and reasoning about a limited domain. Agent's fusion capabilities are based on its local CBN, which captures agent's partial domain expertise.

Definition 2 (Local CBN): A local CBN ψ_i of agent A_i is a tuple $\psi_i = (G_i, \phi(Q_i))$. G_i is a DAG defined on a set of random variables Q_i . In the following text we call Q_i a cluster. The potential $\phi(Q_i)$ is defined according to (see [2]):

$$\phi(Q_i) = \prod_{j=1}^m P(V_j|\pi(V_j)) \quad (4)$$

where variable $V_j \in Q_i$, m is the number of variables defined in the BN ψ_i and where the probability distribution over the service concept $R_i \in Q_i$ is initially set uniform.

Moreover, agent A_i 's fusion task is the computation of a distribution over a single root node corresponding to the service variable $R \in Q_i$. This computation is based on the distributions over the set of input variables $\mathcal{L}_i \subset Q_i$ that correspond to leaf nodes in G_i . The service node is always an ancestor of the input nodes.

Each agent supports local belief propagation as well as fusion of partial beliefs through inter-agent communication.

Local CBN supports also decentralized self organization of DPN agent systems. Namely, variables in the local CBN of a DPN agent encode its local ontology, a set of concepts from the domain that the agent is reasoning about. Note that in every DPN agent the service and input concepts are defined through the labels of variables in local CBNs. Consequently, in this paper a concept is synonymous for a variable.

Local concepts play a central role in the self configuration of meaningful DPN organizations. Namely, two agents can cooperate if the following sharing condition is satisfied:

Definition 3 (Sharing Condition): Agents A_i and A_j can integrate their network fragments if the **service concept** R_i of A_i is identical to an input concept of A_j :

$$\{R_i\} \cap \mathcal{L}_j \neq \emptyset \quad (5)$$

where \mathcal{L}_j is the set of **input concepts** of agent A_j .

In other words, if condition (5) is satisfied the estimated distribution over the service variable R_i can be used as input in agent A_j .

Note that through the local CBNs we specify beforehand *what* types of agents can exchange information. But, we never know in advance *which* agent instances are actually going to participate in the distributed CBN. Also, a single DPN agent can participate in several DPNs accomplishing different fusion tasks simultaneously.

Moreover, for any two DPN agents A_i and A_j satisfying condition 3 we obtain a DPN separator:

Definition 4 (DPN separator): Given clusters Q_i and Q_j the separator $S(Q_i, Q_j)$ is defined by

$$S(Q_i, Q_j) = Q_i \cap Q_j \quad \text{where } |\langle Q_i, Q_j \rangle| = 1 \quad (6)$$

DPN separator is a consequence of the fact that any DPN agent A_i can supply other agents only with a probability distribution over a single variable contained in the cluster Q_i . Clearly, this limits the modeling repertoire to a certain extent. However, this simplification facilitates efficient and robust inference in distributed fusion systems as we will show later.

In general, DPN agents implement distributed retrospective inference based on diagnostic belief propagation (i.e. retrospective support [5]). Namely, service node of an agent is always an ancestor of the input nodes of the same agent. In addition, given the definition of a DPN separator and the fact that agent A_i can supply another agent A_j with a distribution that corresponds to a leaf node in A_j , we see that the inter agent fusion corresponds to the diagnostic reasoning; i.e. the agents implement a reasoning from the symptoms to the causes by reversing the causal relations.

B. DPN Agent Types

A causal model that for a certain estimation time slice correctly maps observations to hypotheses about hidden phenomena, must explicitly capture every observation obtained in that time slice through a node. However, in many real world applications we do not know in advance which information sources will be available. Consequently, the causal model must be modified at runtime. As within a time slice new information sources become available, the corresponding modeling fragments must be appended via propensity

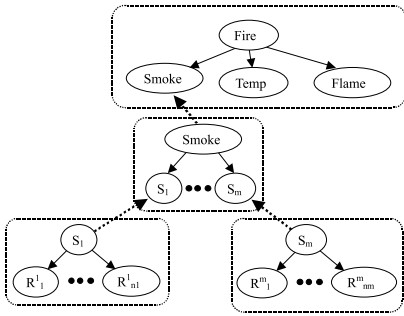


Fig. 2. An organization of DPN agents implementing a distributed causal model. Each dashed rectangle corresponds to an agent with a local CBN. dashed arrows show the direction of the inter-agent fusion flow.

nodes and leaf nodes must be attached to such fragments for each new observation. In other words, a system of DPN agents must cope with the dynamics of the sensing processes as well as changing information source constellations. DPN agents must expand the model at runtime in such a way that the model correctly maps all represented observations to the hypotheses about the hidden phenomena of interest.

We achieve this by introducing three types of DPN agents, each suitable for different modeling aspects. Each agent type is characterized through a set of possible CBN topologies and a specific belief updating algorithm.

Through a distribution of the modeling and fusion capabilities with the help of such building blocks, we can achieve a good modeling flexibility, while the inter-agent belief propagation is very simple and efficient. By combining different types of components we can assemble DPN organizations that implement complex and flexible distributed BNs.

1) *Static Modeling Components*: Agents implementing *Static modeling building blocks* can reason exclusively about distributions over quasi static variables (see section II-B), as for example the top agent in Figure 2. Expertise of a static agent A_i can be captured by a CBN featuring an arbitrarily complex DAG G_i that has a set of quasi-static variables Q_i (i.e. a cluster). The potential over variables \mathcal{V}_i is defined according to definition 2. Only one of the root concepts $\mathcal{R}_i \subseteq \mathcal{V}_i$ of G_i can represent the service concept of agent A_i and its prior probability is uniform. The leaf concepts $\mathcal{L}_i \subseteq \mathcal{V}_i$ of agent A_i correspond to the input concepts.

Arbitrary inference algorithms, such as belief propagation in Junction Trees ([7], [2]), can be deployed. Such algorithms basically carry out multiplication and division with separator potentials and project down the joint distribution to the service variable (see algorithm 1); for a thorough introduction to operations over potential and projections (symbol \downarrow) see [7], [2].

2) *Dynamic Modeling Components*: Agents implementing *dynamic modeling building blocks* can describe sequences of observations, originating from a single information source represented by a sensor propensity node (see agents at the bottom in Figure 2). Since we assume that reports from a single sensor are caused by some hidden quasi-static phenomenon (i.e. sensor's propensity), the sequences

Algorithm 1: Static fusion algorithm

procedure: StaticFusion($\phi'(S\langle Q_i, Q_l \rangle)$)
input : Separator potential $\phi'(S\langle Q_i, Q_l \rangle)$ over a leaf node (corresponding to soft evidence)
output : Distribution $P(R_s)$ over the service root
1 Update potential $\phi(Q_i)$ of graph G_i with the received separator potential $\phi'(S\langle Q_i, Q_l \rangle)$ from agent A_l with variables Q_l with:

$$\phi(Q_i) = \frac{\phi(Q_i)\phi'(S\langle Q_i, Q_l \rangle)}{\phi(S\langle Q_i, Q_l \rangle)} \quad (7)$$

where $\phi(S\langle Q_i, Q_l \rangle)$ is the old separator potential;

- 2 Set $\phi(S\langle Q_i, Q_l \rangle) \leftarrow \phi'(S\langle Q_i, Q_l \rangle)$;
 - 3 compute $\phi(R_s) = \phi(Q_i) \downarrow^{R_s}$;
 - 4 Return $P(R_s) = \alpha \cdot \phi(R_s)$;
-

of resulting observations can be captured by a naive CBN, where each leaf node captures a sensor report. However, reasoning with a naive BN is not practical, since each new observation requires a modification of the local DAG. Instead, we can obtain identical results by using a simple belief updating algorithm 2 that makes use of a CBN consisting of only two nodes. This approach relies on two assumptions: (i) all observations are conditionally independent given the sensor propensity and (ii) the generative model is the same for all observations of a certain type; i.e. all readings of a certain type are obtained through sampling from a model with two nodes.

Algorithm 2: Dynamic fusion algorithm

procedure: DynamicFusion($\phi(e_{E_k})$)
input : Hard evidence encoded in $\phi(e_{E_k})$
output : Distribution $P(R_s)$ over the service root
1 $\phi(R) = P(R_s)P(E_k|R_s)\phi(e_{E_k})$;
2 $P(R) \leftarrow \alpha \cdot \phi(R)$;
3 Return $P(R_s)$;

3) *Appendable Modeling Components*: Agents implementing *appendable modeling building blocks* support expansion of causal models during a time slice as new information sources become available. Recall from section II-B, that with every new information source we introduce a partial causal process, which is reflected in the model by adding a propensity node which in turn is a root node of the leaf nodes corresponding to the observations from that source.

An appendable agent can use local BNs that consist of several conditionally independent nodes given a single variable corresponding to agent's service concept (see the intermediate agent in Figure 2). Due to this limitation, we can use a belief propagation algorithm that can use static networks in conjunction with special data structures in a way that is equivalent to expanding a CBN with a network fragment which is conditionally independent of other network components given some variable. Similarly to the dynamic

fragments, the appendable components use static BNs which encode typical relations between the event types, but do not explicitly model each instance of a newly added variable (e.g. every observation context).

Algorithm 3: Appendable fusion algorithm

procedure: AppendableFusion ($\phi'(S\langle Q_i, Q_k \rangle)$)
input : Separator potential $\phi'(S\langle Q_i, Q_k \rangle)$ over a leaf node (corresponding to soft evidence)
output : Distribution $P(R_s)$ over the service root

1

$$\phi'_k(R_s) \leftarrow \left(P(R_s) \sum_{L_k} P(L_k | R_s) \phi'(S\langle Q_i, Q_k \rangle) \right) \downarrow_{R_s}$$

if $S\langle Q_i, Q_k \rangle$ is from a new information source **then**

2 $\phi_\Psi(R_s) \leftarrow \phi_\Psi(R_s) \phi'_k(R_s)$;

3 Append uniform potential $\phi_k(R_s)$ to a list of separator potentials of all leaf nodes;

4 **else**

5 $\phi_\Psi(R_s) \leftarrow \frac{\phi_\Psi(R_s) \phi'_k(R_s)}{\phi_k(R_s)}$;

6 **end**

7 Set $\phi_k(R_s) \leftarrow \phi'_k(R_s)$;

8 Return $P(R_s) = \alpha \cdot \phi_\Psi(R_s)$;

C. DPN Fusion Organization

Each DPN fusion process depends on the constellation of cooperating agents, which corresponds to a particular Problem/Task decomposition. In this context we introduce the concept of a DPN Fusion Organization that can be formalized through a function $\mathcal{F}(q, \mathcal{A}) : \mathcal{A}, q \rightarrow \Omega$, which maps the set of existing agents \mathcal{A} and a query concept q to a graph Ω . Ω is a tuple $\Omega = \langle \mathcal{A}_q, \mathcal{R}_q \rangle$, where $\mathcal{A}_q \subseteq \mathcal{A}$ denotes the set of agents that can provide information relevant for the reasoning about the distribution over the variable corresponding to the query concept q . \mathcal{R}_q represents a set of agent pairs that satisfy condition (5), i.e. can share partial beliefs about the states of variables.

Clearly, a particular DPN fusion organization reflects local agent ontologies and it determines which types of information can be processed and in what order this can be done.

In other words, a fusion organization corresponds to assembled probabilistic causal models, which must support mathematically rigorous as well as efficient information fusion. DPNs often consist of large numbers of different DPN fusion agents processing large amounts of heterogeneous information. In such systems huge quantities of evidence are used to instantiate nodes contained in agents that in turn correspond to different nodes in a DPN organization Ω . Consequently, it is difficult to obtain a central overview of the instantiated variables, especially if the instantiations are frequent. Similarly, it can be very difficult to obtain a centralized overview over all local BNs participating in a particular DPN organization, since DPN fusion organizations can change their topology at runtime as new agents join or

leave an organization during a fusion process. Therefore, standard approaches to belief propagation turn out to be impractical, since they require preprocessing, such as construction of Junction Trees, loop-cutsets (see [5]), etc., which in general require knowledge of all participating local BNs and partial synchronization of processes.

By introducing a few design constraints we can obtain distributed models that do not require any preprocessing of the assembled BNs and exact inference can be achieved without any synchronization of the belief propagation processes distributed through different agents.

Definition 5 (DPN Organization Constraints): Assume a DPN organization Ω of agents corresponding to a set of clusters $\Gamma = \{Q_1, \dots, Q_n\}$ and a set of separators $\mathcal{S} = \{S_1, \dots, S_n\}$ between these clusters. Each cluster $Q_j \in \Gamma$ contains all variables from the local CBN supported by agent A_j in a particular DPN organization. A new agent A_i can join this DPN organization Ω if its cluster Q_i satisfies the following conditions:

- (i) $\exists! Q_j (Q_i \cap Q_j) \neq \emptyset$
- (ii) $|Q_i \cap Q_j| = |S\langle Q_i, Q_j \rangle| = 1$

Condition (i) captures the *intersection constraint* and states that in a particular DPN organization there can exist only one cluster $Q_j \in \Gamma$ for which the newly added cluster Q_i has a non-empty intersection set. In other words, given the joining agent A_i there exists exactly one agent A_j , such that the Integration condition is satisfied (see Definition 3). Condition (ii), on the other hand, captures the *separator constraint* and states that all separators must have size 1; i.e. contain a single random variable (see Definition 4).

Only agents containing BNs for which the clusters satisfy both constraints simultaneously can join a DPN organization. Consequently, each DPN organization is associated with a specific distributed domain model:

Definition 6 (DPN Domain Model): A DPN domain model Ψ is defined as a tuple $D = (\mathcal{G}, \mathcal{V}, R, \mathcal{P})$. \mathcal{G} is the set of local DAGs of the all DPN agents participating in a particular fusion organization. \mathcal{V} is a union of all variables from local clusters and $Q_h \in \mathcal{V}$ is the cluster that contains the hypothesis node. R is the set of DPN separators where every separator $S_i \in R$ is defined according to definition 4. For every pair of separators $S_i \in R$ and $S_j \in R$ the intersection $S_i \cap S_j = \emptyset$ where $i \neq j$. \mathcal{P} is the set of potentials defined over the DPN domain model and $P_i \in \mathcal{P}$ are the potentials for cluster Q_i (see definition 2).

Note that because of the design constraints (5), a DPN domain model always corresponds to a DPN organization Ω (i.e. a graph) featuring tree topologies.

Given the DPN domain model definition, it is obvious that DPN systems are best suited for domains which can be described through BNs featuring topologies with a significant portion of conditionally independent network fragments. Despite the limitation to relatively simple topologies we can describe a significant class of relevant non-deterministic domains through such models.

D. Distributed Inference

Belief propagation in a DPN organization can be viewed as a combination of local fusion processes that are implemented by the three DPN agent types and run simultaneously on networked devices. By introducing the design constraints (5) and the three DPN agent types, we can show that fusion in DPNs is (i) independent of the order of evidence instantiations throughout of a set of agents, (ii) it does not require any centralized fusion control, (iii) can efficiently cope with changing networks at runtime and (iv) does not require any preprocessing, such as compilation of junction trees or loop cut-sets, which can be computationally expensive and might require partial synchronization.

After the full evidence set \mathcal{E} is propagated through a system of fragments in a DPN the computed marginal posterior $P(\text{Gas}X|\mathcal{E})$ will reflect entire evidence set \mathcal{E} correctly.

Proposition 1 (Correct Posterior Calculation): Let us assume a DPN Ψ that computes belief over hypothesis variable H and set of evidence \mathcal{E} that is used for the instantiation of input variables in different agents of DPN Ψ . If the dynamic modeling fragments are used only in agents that have direct access to subsets of evidence in \mathcal{E} , then after execution of algorithms 1, 2 and 3 on different agents will result in a posterior $P(H|\mathcal{E})$ which will correctly reflect the entire evidence set \mathcal{E} . This distributed process is equivalent to exact belief propagation in the monolithic CBN that captures the complete causal process.

Proof: (sketch) If we investigate the relations between the CBN graphical models and the corresponding factorizations we can observe that the graphical model, modeling a sequence of events, corresponds to a nested factorization. Every factor is defined by a summation over factors. For every factor in this summation we can again identify another nested factorization, etc. Therefore, the innermost factors will correspond to the leaf nodes of the sequence of modeled events where evidence is instantiated. The inference is based on the reverse causal direction between local CBN and therefore the agents automatically combine different factors, that will correspond to partial fusion results, in such an order that the nested factorization is correct. In other words the encoded conditional independence given the service concept in the graphical model is directly related to the factorization order. \square

IV. SELF ORGANIZATION PRINCIPLES

Each fusion task, i.e. reasoning about a particular hypothesis, requires a specific world model that maps the currently available evidence from the relevant information sources to the hypothesis. The model must capture every piece of evidence by a separate node. Since we assume that the information sources are not known prior to the operation, we must be able to generate such models by using smaller building blocks at runtime. Given the fact that each agent provides a modeling building block, a local CBN, DPN Fusion Organizations play a crucial role. Namely, a DPN Fusion Organization describes how different predefined modeling building blocks are combined into a meaningful

distributed world model that correctly maps heterogeneous evidence to a distribution over a hypothesis variable.

In other words, self configuration must result in appropriate DPN Fusion Organizations. In addition, we require that DPN Fusion Organizations implement distributed BNs that correspond to DPN Domain Models; i.e. the organization constraints (5) must be satisfied.

By using local causal models distributed throughout different agents, we can derive algorithms which support self-organization based on the cooperation between individual agents without any centralized configuration control. The basic self-organization mechanism is implemented through the algorithm for a Top-Down Network Configuration (see algorithm 4). Essentially, this algorithm implements a simple configuration rule: *After some agent A has become a member of a particular DPN, it starts looking for other agents whose output concepts correspond to its input concept.*

Algorithm 4: Top Down Network Configuration

```

procedure: TopDownConfiguration( $X$ )
input      :  $X$  is a query concept
1 Find a set of agents  $\mathcal{A}_q \subseteq \mathcal{A}$  such that
   $\forall A_i \in \mathcal{A}_q : R_i = X$ ;
2 foreach agent  $A_i \in \mathcal{A}_q$  do
3   Use Contract Net Protocol to establish a fusion
   contract with  $A_i$  by satisfying the organization
   constraints in definition 5;
4   if fusion contract with  $A_i$  is established successfully
   then
5     foreach input concept  $L_{i,j} \in \mathcal{L}_i$  from agent  $A_i$ 
     do
6        $A_i$  calls:
       TopDownConfiguration( $L_{i,j}$ );
7     end
8   end
9 end

```

Algorithm 4 is suitable for the situations where we can provide a query concept to a set of agents, which respond and organize meaningful DPN fusion systems that integrate all relevant agents and information sources for a given fusion task. Algorithm 4 implements the Contract Net Protocol.

However, often it is desirable to organize fusion systems in response to unusual observations. In other words, DPNs can serve as alarm systems. Therefore, DPN agents also implement a protocol that supports discovery of all possible DPN organizations that would make use of certain types of observations. This protocol is captured by the algorithm for a Bottom-Up self configuration.

By using this algorithm, we can implement robust alarm-ing systems. A single sensor observation can activate many DPN fusion organizations simultaneously. However, each of these organizations tries to find as many other relevant information sources as possible. In this way a false alarm by one sensor can effectively be overridden through the reports from other information sources, if a sufficient portion of

Algorithm 5: Bottom-Up Network Configuration

procedure: BottomUpConfiguration(R_k)**input** : R_k is the service concept of the caller agent A_k

- 1 Find a set of agents $\mathcal{A}_s \subseteq \mathcal{A}$ such that $\forall A_i \in \mathcal{A}_s : R_k \in \mathcal{L}_i$;
 - 2 **foreach** agent $A_i \in \mathcal{A}_s$ **do**
 - 3 **foreach** input concept $L_{i,j} \in \mathcal{L}_i$ from agent A_i **do**
 - 4 A_i calls: TopDownConfiguration($L_{i,j}$);
 - 5 **end**
 - 6 A_i calls: BottomUpConfiguration(R_i);
 - 7 **end**
-

information sources work properly. The decision maker is then presented with different hypotheses corresponding to critical events if their probabilities exceed certain predefined thresholds. Algorithm 5 combines algorithm 4 with a bottom-up messaging protocol.

Algorithms 4 and 5 implement concept driven configuration processes, which exploit the fact that causal links, which should be established between the partial world models, correspond to the intersections of local ontologies capturing service and input concepts. Self configuration processes support a great flexibility through adaptive combination of standard building blocks. The configuration processes are decentralized, since agents initiate cooperation pairwise, by using their local ontologies encoded by their local world models. In addition, these algorithms enforce a partial order of possible link creations. Thus, by using the presented algorithms we can build hierarchical fusion systems through a hierarchical service discovery.

Also, given the constraints for local BNs in DPN agents, we can easily show that configuration algorithms 4 and 5 automatically construct DPN fusion organizations that implement DPN Domain Graphs.

V. DISCUSSION

We presented DPNs, fusion systems that are based on causal models. Such models facilitate the design of flexible modeling components as well as self-configuration of meaningful fusion systems. We show that by explicitly considering causal models we can obtain distributed fusion systems featuring:

- Distributed computation of posterior distribution over a single hypothesis variable, which correctly reflects the entire evidence set that has been injected through different parts (i.e. agents) of a DPN system.
- Exact distributed belief propagation without compilation of global fusion structures. Only local models are compiled independently prior to runtime.
- Fusion results are correct without using any control and synchronization of distributed partial fusion processes.

In contrast to more general approaches to distributed fusion, such as [7], information fusion in DPNs is based on

retrospective inference with BNs featuring limited topologies. While this limitation reduces the modeling repertoire of the DPN systems, they are relevant for a significant class of situation assessment problems, where the relevant information sources must quickly be integrated at runtime and the reasoning is based on large quantities of heterogeneous and noisy information. In addition, DPNs support automated assembly of meaningful distributed BNs as well as a fully decentralized distributed belief propagation. This makes DPNs very robust and suitable for applications, where the constellations of information sources as well as agent systems change frequently at runtime. Note that the more general approach described in [7] is not suitable for such dynamic settings.

Also, there are approaches that make use of Bayesian decentralized fusion for tracking applications, such as [3]. However, these methods focus primarily on the fusion of rather homogeneous information, such as location of objects. The DPN, on the other hand, can fuse semantically very heterogeneous information types. In addition, the DPN architecture can be used in a tracking process, where the hypothesis variable is evolving through a series of timeslices according to some first order Markov process. In other words, DPNs can implement very complex sensor models that are used for the advanced estimation steps in a filtering process based on very heterogeneous observations.

Also, the concept of sensor propensity might look impractical, since it can be very difficult to obtain modeling parameters that would precisely describe the true distributions over the combinations of the propensity states and states of the related phenomena. However, recently it has been shown that given BNs with high branching factors, fusion can be very reliable with very different modeling parameters [4]; the parameters must merely capture very simple greater/smaller than relations between the probabilities in the true distributions, which can easily be identified by experts or machine learning algorithms. With the help of the Inference Meta Model we can show that under certain circumstances fusion in DPN systems is analogous to repetition coding technique and its estimation accuracy can be very high despite significant discrepancies between the true and the modeled conditional probability distributions (i.e. CPTs).

REFERENCES

- [1] P. de Oude, B. Ottens, and G. Pavlin. Information fusion in distributed probabilistic networks. In *Artificial Intelligence and Applications*, pages 195–201, Innsbruck, Austria, 2005.
- [2] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.
- [3] Alexei Makarenko and Hugh F. Durrant-Whyte. Decentralized data fusion and control in active sensor networks. In *Proceedings of the Seventh International Conference on Information Fusion*, Jun 2004.
- [4] G. Pavlin and J. Nunnink. Inference meta models: Towards robust information fusion with bayesian networks. In *to appear in Proc. Int. Conf. on Information Fusion*, pages 846–852, Florence, Italy, 2006.
- [5] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [6] Judea Pearl. Graphs, causality, and structural equation models. Technical Report 980004, 31, 1998.
- [7] Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.